

# RotationNet: Joint Learning of Object Classification and Viewpoint Estimation using Unaligned 3D Object Dataset

Asako Kanezaki<sup>\*1</sup>, Yasuyuki Matsushita<sup>†2</sup> and Yoshifumi Nishida<sup>‡1</sup>

<sup>1</sup>National Institute of Advanced Industrial Science and Technology (AIST)

<sup>2</sup>Graduate School of Information Science and Technology, Osaka University

## Abstract

We propose a Convolutional Neural Network (CNN)-based model “RotationNet,” which takes multi-view images of an object as input and estimates both its pose and object category. Unlike previous approaches that use known pose labels for training, our method treats the pose labels as latent variables, which are optimized to self-align in an unsupervised manner during the training using an unaligned dataset. RotationNet is designed to use only a partial set of multi-view images for inference, and this property makes it useful in real-world scenarios where only partial views are available. Moreover, our pose alignment strategy enables one to obtain view-specific feature representations shared across classes, which is important to maintain high accuracy both in object categorization and pose estimation. Effectiveness of RotationNet is demonstrated by its comparable performance to the state-of-the-art methods of 3D object classification on 10- and 40-class object benchmark datasets. We also show that RotationNet, with only a few partial views, achieves comparable performance to the multi-view CNN method that uses 80 different view images. The code is available on <https://github.com/kanezaki/rotationnet>.

## 1. Introduction

Object classification accuracy can be enhanced by the use of multiple different views of a target object, which is known as an “active recognition” approach [3, 16]. Recent remarkable advances in image recognition and collection of 3D object models enabled the learning of multi-view representations of objects in various categories. However, in real-world scenarios, objects can often only be observed from

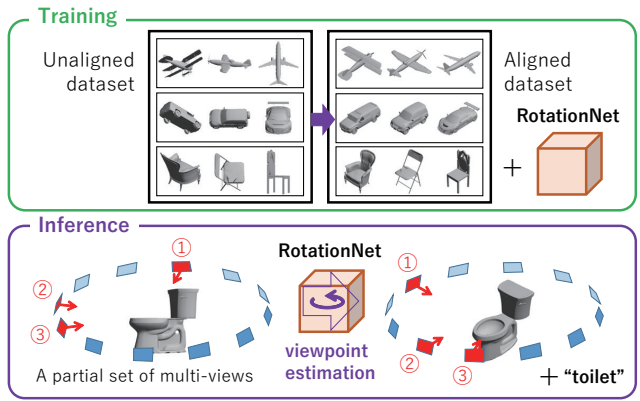


Figure 1. Overview of training and inference using RotationNet. We jointly learn the parameters of RotationNet and estimate the pose of object instances in the unaligned 3D object dataset. In the inference phase, RotationNet takes a partial set of all the multi-view images of an object as input and outputs its object category by rotation, where the best pose is selected to maximize the object category likelihood.

limited viewpoints due to occlusions, which makes it difficult to rely on multi-view representations that are learned with the whole circumference. It has been understood that if the viewpoint is known the object classification accuracy can be improved. Likewise, if the object category is known, that helps infer the viewpoint. As such, object classification and viewpoint estimation is a tightly coupled problem, which can best benefit from their joint optimization.

We propose a new Convolutional Neural Network (CNN) model that we call “RotationNet,” which takes multi-view images of an object as input and outputs its pose and object category. The overview of training and inference using RotationNet is illustrated in Fig. 1. Whereas RotationNet requires a complete set of multi-view images of an object captured from all the pre-defined viewpoints in the training process, it is able to take only a partial set of all the

<sup>\*</sup>kanezaki.asako@aist.go.jp

<sup>†</sup>yasumat@ist.osaka-u.ac.jp

<sup>‡</sup>y.nishida@aist.go.jp

multi-view images – a single image at minimum – as input for inference. This property is particularly important for object classification in real-world scenarios, where it is often difficult to place a camera at all the predefined viewpoints due to occlusions. In addition, RotationNet does not require the multi-view images to be provided at one time but allows their sequential input and updates of the target object’s category likelihood. This property is suitable for applications that require on-the-fly classification with a moving camera.

The most characteristic feature of RotationNet is that it treats view labels associated with images as *latent* variables during the training. This enables unsupervised learning of object poses using an unaligned object dataset; thus, it eliminates the need of preprocessing for pose normalization that is often sensitive to noise and individual differences in shape. Our method automatically determines the basis axes of objects based on their appearance during the training and achieves not only intra-class but also inter-class object pose alignment. Inter-class pose alignment is important to deal with joint learning of object pose and category, because it may become an ill-posed problem to obtain a model to distinguish, *e.g.*, a car and a bus if the side view of a car is compared with the frontal view of a bus.

We show that RotationNet achieves the second-best classification performance on 3D object benchmark datasets consisting of 10- and 40-categories, the best among the methods that do not use an ensembling approach. In addition, we show that our model generalizes well to a real-world image dataset that was newly created for the general task of multi-view object classification.<sup>1</sup>

## 2. Related work

There are two main approaches for the CNN-based 3D object classification: voxel-based and 2D image-based approaches. The earliest work on the former approach is 3D ShapeNets [26], which learns a Convolutional Deep Belief Network that outputs probability distributions of binary occupancy voxel values. There are also other voxel-based CNN approaches such as VoxNet [15], which consists of two convolutional layers connected to one pooling layer and two fully-connected layers, and VRN [4], which is based on the latest ResNet architecture [7]. Even when working with 3D objects, 2D image-based approaches are shown effective for general object recognition tasks. Su *et al.* [23] proposed multi-view CNN (MVCNN), which takes multi-view images of an object captured from surrounding virtual cameras as input and outputs the object’s category label. Multi-view representations are also used for 3D shape retrieval [1] or used together with voxel representations for object classification [8]. Qi *et al.* [17] gives a comprehensive study on the voxel-based CNNs and multi-view CNNs for 3D object

classification.

Because MVCNN integrates multi-views in a view-pooling layer which lies in the middle of the CNN, it requires a complete set of multi-view images recorded from all the predefined views for object inference. Unlike MVCNN, our method is able to classify an object using a partial set of multi-view images that may be sequentially observed (Fig. 2). Elhoseiny *et al.* [6] recently explored CNN architectures for joint object classification and pose estimation learned with multi-view images. Whereas their method takes a single image as input for its prediction, we aggregate predictions from multiple images captured from different viewpoints.

Viewpoint estimation is significant in its role in improving object classification. Better performance was achieved on face identification [28] and human action classification [5] by generating unseen views after observing a single view. Su *et al.* [24] synthesized HOG features of different views of an object to obtain a view-independent feature representation. These methods observe a single image and “imagine” the appearance of objects’ unobserved profiles, which is innately more uncertain than using real observations. To utilize multi-view images for 3D object classification, Johns *et al.* [10] proposed a method to learn pairs of images captured from different viewpoints and produce the best camera trajectory for sequential object observation. Sedaghat *et al.* [20] proposed a voxel-based CNN that outputs orientation labels as well as classification labels and demonstrated that it improved 3D object classification performance.

Although joint learning of object classification and pose estimation has been widely studied [19, 14, 27, 2], inter-class pose alignment has drawn little attention. However, it is beneficial to share view-specific appearance information across classes to simultaneously solve for object classification and pose estimation. Kuznetsova *et al.* [12] pointed out this issue and presented a metric learning approach that shares visual components across categories for simultaneous pose estimation and class prediction. Our method also uses a model with view-specific appearances that are shared across classes; thus, it is able to maintain high accuracy for both object classification and pose estimation.

All the methods mentioned above assume known poses in training samples; however, object poses are not always aligned in existing object databases. Instead of applying pose normalization such as those based on principal component analysis (PCA), our method aligns object poses via unsupervised viewpoint estimation. In such a perspective, our method is related to spatial transformer networks [9], which learned networks that predict translation, scale, and rotation of an object in an image. This method uses only category labels of training samples to optimize the parameters for the spatial transformation. While it focuses on 2D

<sup>1</sup>The dataset will be available soon.

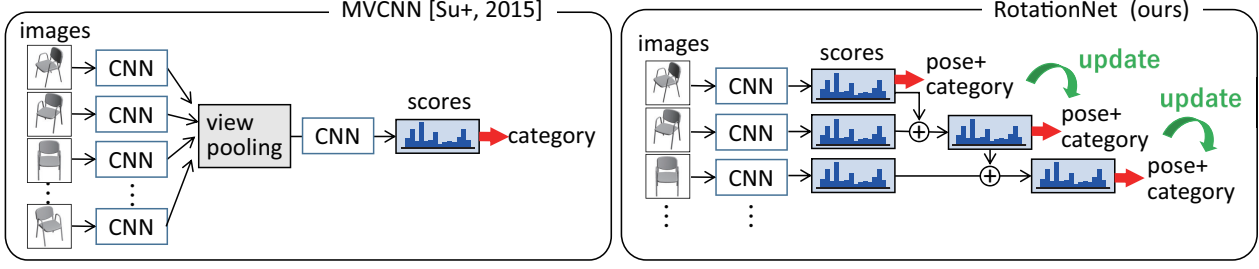


Figure 2. Illustration of the difference between the testing processes of MVCNN [23] and RotationNet. Unlike MVCNN, our method is designed to classify an object using a partial set of multi-view images that may be sequentially observed.

spatial transformation, our work aims at determining object rotation in 3D space.

### 3. Proposed method

The training process of RotationNet is illustrated in Fig. 3. We assume that multi-view images of each training object instance are observed from all the pre-defined view-points. Let  $M$  be the number of the pre-defined viewpoints and  $N$  denote the number of target object categories. A training sample consists of  $M$  images of an object  $\{\mathbf{x}_i\}_{i=1}^M$  and its category label  $y \in \{1, \dots, N\}$ . We attach a view label variable  $v_i \in \{1, \dots, M\}$  to each image  $\mathbf{x}_i$  and set it to  $j$  when the image is observed from the  $j$ -th viewpoint, i.e.,  $v_i \leftarrow j$ . In our method, only the category label  $y$  is given during the training whereas the view labels  $\{v_i\}$  are unknown, namely,  $\{v_i\}$  are treated as latent variables that are optimized in the training process.

RotationNet is defined as a differentiable multi-layer neural networks  $R(\cdot)$ . The final layer of RotationNet is the concatenation of  $M$  softmax layers that output  $P(\hat{y}_i | \mathbf{x}_i, v_i = j)$  ( $j = 1, \dots, M$ ) for each image  $\mathbf{x}_i$ . Here,  $\hat{y}_i$  denotes an estimate of the object category label for  $\mathbf{x}_i$ . For the training of RotationNet, we input the set of images  $\{\mathbf{x}_i\}_{i=1}^M$  simultaneously and solve the following optimization problem:

$$\max_{R, \{v_i\}_{i=1}^M} \prod_{i=1}^M P(\hat{y}_i = y | \mathbf{x}_i, v_i). \quad (1)$$

The parameters of  $R$  and latent variables  $v_i$  are optimized to output the highest probability of  $y$  for the input of multi-view images  $\{\mathbf{x}_i\}_{i=1}^M$ .

Now, we describe how we design  $P(\hat{y}_i | \mathbf{x}_i, v_i)$  outputs. First of all, the posterior  $P(\hat{y}_i = y | \mathbf{x}_i, v_i)$  should become close to one when  $v_i$  is correct. Otherwise, in the case that  $v_i$  is incorrect,  $P(\hat{y}_i = y | \mathbf{x}_i, v_i)$  may not necessarily be high because the image  $\mathbf{x}_i$  is captured from a different viewpoint. As described above, we decide the view labels  $\{v_i\}_{i=1}^M$  according to the  $P(\hat{y}_i = y | \mathbf{x}_i, v_i)$  outputs as in (1). In order to obtain a stable solution of (1), we introduce

an ‘‘incorrect view’’ label and append it to the target category labels. The ‘‘incorrect view’’ label plays a similar role to the ‘‘background’’ label for object detection tasks, which represents negative samples that belong to no class. Then, RotationNet calculates  $P(\hat{y}_i | \mathbf{x}_i, v_i)$  by applying softmax functions to the  $(N + 1)$ -dimensional outputs, where  $\sum_{\hat{y}_i=1}^{N+1} P(\hat{y}_i | \mathbf{x}_i, v_i) = 1$ . Here,  $P(\hat{y}_i = N + 1 | \mathbf{x}_i, v_i)$  indicates how likely it is that the view label  $v_i$  is incorrect.

Based on the above discussion, we substantiate (1) as follows. Letting  $P_i = \{p_{j,k}^{(i)}\} \in \mathbb{R}_+^{M \times (N+1)}$  denote a matrix composed of  $P(\hat{y}_i | \mathbf{x}_i, v_i)$  for all the  $M$  viewpoints and  $N + 1$  classes, the target value of  $P_i$  in the case that  $v_i$  is correctly estimated is defined as follows:

$$p_{j,k}^{(i)} = \begin{cases} 1 & (j = v_i \text{ and } k = y), \text{ or} \\ & (j \neq v_i \text{ and } k = N + 1) \\ 0 & (\text{otherwise}). \end{cases} \quad (2)$$

In this way, (1) can be rewritten as the following cross-entropy optimization problem:

$$\max_{R, \{v_i\}_{i=1}^M} \sum_{i=1}^M \left( \log p_{v_i, y}^{(i)} + \sum_{j \neq v_i} \log p_{j, N+1}^{(i)} \right). \quad (3)$$

If we fix  $\{v_i\}_{i=1}^M$  here, the above can be written as a sub-problem of optimizing  $R$  as follows:

$$\max_R \sum_{i=1}^M \left( \log p_{v_i, y}^{(i)} + \sum_{j \neq v_i} \log p_{j, N+1}^{(i)} \right), \quad (4)$$

where the parameters of  $R$  can be iteratively updated via standard back-propagation of  $M$  softmax losses. Since  $\{v_i\}_{i=1}^M$  are not constant but latent variables that need to be optimized during the training of  $R$ , we employ alternating optimization of  $R$  and  $\{v_i\}_{i=1}^M$ . More specifically, in every iteration, our method determines  $\{v_i\}_{i=1}^M$  according to  $P_i$  obtained via forwarding of (fixed)  $R$ , and then update  $R$  according to the estimated  $\{v_i\}_{i=1}^M$  by fixing them.

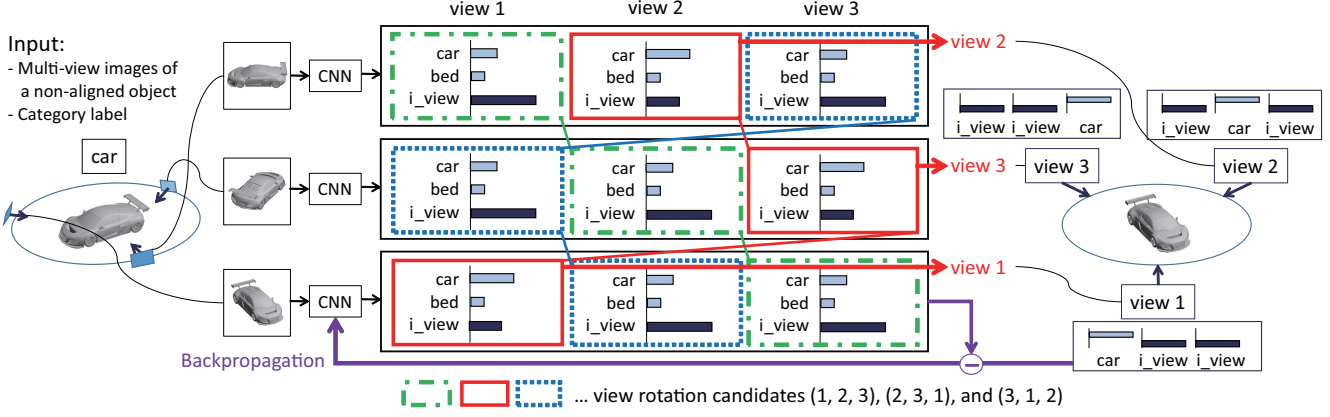


Figure 3. Illustration of the training process of RotationNet, where the number of views  $M$  is 3 and the number of categories  $N$  is 2. A training sample consists of  $M$  images of an unaligned object instance and its category label  $y$ . For each input image, our CNN (RotationNet) outputs  $M$  histograms with  $N + 1$  bins whose norm is 1. The last bin of each histogram represents the “incorrect view” class, which serves as a weight of how likely the histogram does not correspond to each view label. According to the histogram values, we decide which image corresponds to views 1, 2, and 3. There are three candidates for view rotation: (1, 2, 3), (2, 3, 1), and (3, 1, 2). For each candidate, we calculate the score for the ground-truth category (“car” in this case) by multiplying the histograms and selecting the best choice: (2, 3, 1) in this case. Finally, we update the CNN parameters in a standard back-propagation manner with the estimated view labels.

The latent view labels  $\{v_i\}_{i=1}^M$  are determined by solving the following problem:

$$\begin{aligned}
 & \max_{\{v_i\}_{i=1}^M} \sum_{i=1}^M \left( \log p_{v_i, y}^{(i)} + \sum_{j \neq v_i} \log p_{j, N+1}^{(i)} \right) \\
 &= \max_{\{v_i\}_{i=1}^M} \sum_{i=1}^M \left( \log p_{v_i, y}^{(i)} + \sum_{j=1}^M \log p_{j, N+1}^{(i)} - \log p_{v_i, N+1}^{(i)} \right) \\
 &= \max_{\{v_i\}_{i=1}^M} \prod_{i=1}^M \frac{p_{v_i, y}^{(i)}}{p_{v_i, N+1}^{(i)}}, \quad (5)
 \end{aligned}$$

in which the conversion used the fact that  $\sum_{j=1}^M \log p_{j, N+1}^{(i)}$  is constant w.r.t.  $\{v_i\}_{i=1}^M$ . Because the number of candidates for  $\{v_i\}_{i=1}^M$  is limited, we calculate the evaluation value of (5) for all the candidates and take the best choice.

In the inference phase, RotationNet takes as input  $M'$  ( $1 \leq M' \leq M$ ) images of a test object instance, either simultaneously or sequentially, and outputs  $M'$  probabilities. Finally, it estimates the category of the object and the pose labels as follows:

$$\left\{ \hat{y}, \{\hat{v}_i\}_{i=1}^{M'} \right\} = \arg \max_{y, \{v_i\}_{i=1}^{M'}} \prod_{i=1}^{M'} \frac{p_{v_i, y}^{(i)}}{p_{v_i, N+1}^{(i)}}. \quad (6)$$

Similarly to the training phase, we decide  $\{\hat{v}_i\}_{i=1}^{M'}$  according to the outputs  $\{P_i\}_{i=1}^{M'}$ . Thus RotationNet is able to estimate the pose of the object as well as its category label.

**Viewpoint setups for training** While choices of the view labels  $\{v_i\}_{i=1}^{M'}$  can be arbitrary, we consider two setups in

this paper, with and without an upright orientation assumption, similarly to MVCNN [23].

**Case (i): with upright orientation** In the case where we assume upright orientation, we fix a specific axis as the rotation axis (e.g., the  $z$ -axis), which defines the upright orientation, and then place virtual cameras at intervals of the angle  $\theta$  around the axis, elevated by  $\phi$  (set to  $30^\circ$  in this paper) from the ground plane. We set in this paper  $\theta = 30^\circ$ , which yields 12 views for an object ( $M = 12$ ). Here, we define that “view  $m + 1$ ” is obtained by rotating the view position “view  $m$ ” by the angle  $\theta$  about the  $z$ -axis. Note that the view obtained by rotating “view  $M$ ” by the angle  $\theta$  about the  $z$ -axis corresponds to “view 1.” We assume the sequence of input images is consistent with respect to a certain direction of rotation in the training phase. For instance, if  $v_i$  is  $M'$  ( $M' < M$ ), then  $v_{i+1}$  is  $M' + 1$ . Thus the number of candidates for all the view labels  $\{v_i\}_{i=1}^M$  is  $M$ .

**Case (ii): w/o upright orientation** In the case where we do not assume upright orientation, we place virtual cameras on the  $M = 20$  vertices of a dodecahedron encompassing the object. This is because a dodecahedron has the largest number of vertices among regular polyhedra, where view-points can be completely equally distributed in 3D space. Unlike case (i), where there is a unique rotation direction, there are three different patterns of rotation from a certain view, because three edges are connected to each vertex of a dodecahedron. Therefore, the number of candidates for all the view labels  $\{v_i\}_{i=1}^M$  is  $60 (= 3M)^2$ .

<sup>2</sup>In other words, a dodecahedron has 60 rotational (or orientation-preserving) symmetries.

## 4. Experiments

In this section, we show the results of the experiments with 3D model benchmark datasets (Sec. 4.1), a real image benchmark dataset captured with a one-dimensional turning table (Sec. 4.2), and our new dataset consisting of multi-view real images of objects viewed with two rotational degrees of freedom (Sec. 4.3). The architecture of our CNN is based on AlexNet [11], which is smaller than the VGG-M network architecture that MVCNN [23] used. To train RotationNet, we fine-tune the weights pre-trained using the ILSVRC 2012 dataset [18]. The fine-tuning of our CNN converged within 30,000 iterations, which takes approximately 24 hours using a single Geforce GTX TITAN X. We used classical momentum SGD with a learning rate of 0.0005 and a momentum of 0.9 for optimization.

As a baseline method, we also fine-tuned the pre-trained weights of a standard AlexNet CNN that only predicts object categories. To aggregate the predictions of multi-view images, we summed up all the scores obtained through the CNN. This method can be recognized as a modified version of MVCNN [23], where the view-pooling layer is placed after the final softmax layer. We chose average pooling for the view-pooling layer in this setting of the baseline, because we observed that the performance was better than that with max pooling. We also implemented MVCNN [23] based on the AlexNet architecture with the original view-pooling layer for a fair comparison. For the implemented MVCNN and RotationNet, we altered the batch size with 120, 240, 480, 768 (for case (i)) and 900 (for case (ii)) and chose the best one.

### 4.1. Experiment on 3D model datasets

In this section, we describe the experimental results on two 3D model benchmark datasets, ModelNet10 and ModelNet40 [26]. ModelNet10 consists of 4,899 object instances in 10 categories, whereas ModelNet40 consists of 3,983 object instances in 40 categories. We use the same training and test split of ModelNet40 as in [26] and [23]. We prepared multi-view images (i) with the upright orientation assumption and (ii) without the upright orientation assumption using the rendering software published in [23].

Figures 4 and 5 show the state transition of the inter- and intra-class object pose alignment that is automatically achieved during the training of RotationNet with ModelNet40, which depict the variation of the average images generated by concatenating multi-view images in order of their predicted view labels. The figures correspond to cases (i) and (ii), respectively. We can see that the variance of average images decreases together with the variance of object poses. The red dotted lines show the mean variance of average images of all the 40 classes, whereas the blue lines show the variance of average images of the “chair” class.

The images of test object instances in the “chair” class

with the same predicted view label are shown in the top right corners of the figures. In both cases (i) and (ii), where the latter case is more interesting, the chairs with initial random poses gradually get aligned in the same direction after several hundreds of training iterations. Moreover, the average images of all the 40 classes with the same predicted view label shown in red boxes indicate that not only the intra-class alignment but also the inter-class alignment is achieved. The alignment is less obvious in the red boxes of Fig. 5; however, it is confirmed that this does not harm the object classification accuracy.

Next, we show the change of object classification accuracy versus the number of views used for prediction in cases (i) and (ii) with ModelNet40 and ModelNet10, respectively, in Fig. 6 (a)-(b) and Fig. 7 (a)-(b). Here, we show the average scores of 120 trials with randomly selected multi-view sets. We show the results of RotationNet with red lines, those of the baseline method that applies late fusion of multi-view scores with green lines, and those of our implementation of MVCNN using rectangular markers. In Figs. 6 (a) and 7 (a), which show the results with ModelNet40, we also draw the scores with the original MVCNN using Support Vector Machine (SVM) reported in [23]. Interestingly, as we focus on the object classification task whereas Su *et al.* [23] focused more on object retrieval task, we found that the baseline method with late view-pooling is slightly better in this case than the original MVCNN with the view-pooling layer in the middle. The baseline method does especially well with ModelNet10 in case (i) (Fig. 6 (b)), where it achieves the best performance among the methods. With ModelNet40 in case (i) (Fig. 6 (a)), RotationNet achieved a comparable result with MVCNN when we used all the 12 views as input. In case (ii) (Figs. 7 (a) and (b)), where we consider full 3D rotation, RotationNet demonstrated superior performance to other methods. Only with three views, it showed comparable performance to that of MVCNN with a full set of multi-view images.

Finally, we summarize the comparison of classification accuracy on ModelNet40 and ModelNet10 to existing 3D object classification methods in Table 1. RotationNet achieves the second-best place with both the ModelNet40 and ModelNet10 datasets, which is the best among the methods that do not take advantage of ensembling.

### 4.2. Experiment on a real image benchmark dataset

Now we describe the experimental results on a benchmark RGBD dataset published in [13], which consists of real images of objects on a one-dimensional rotation table. This dataset contains 300 object instances in 51 categories. Although it contains depth images and 3D point clouds, we used only RGB images in our experiment. We applied the upright assumption (case (i)) in this experiment, because the bottom faces of objects on the turning table were not



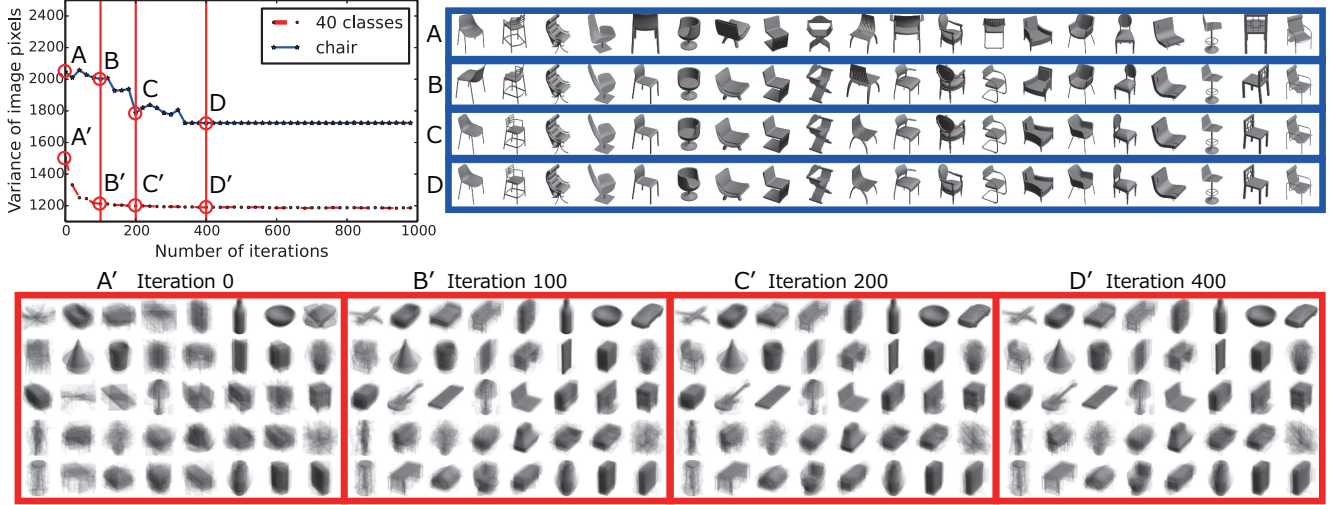


Figure 4. Variance change of the average images generated by concatenating multi-view images (case (i)) in order of their predicted view labels. Average images of 40 classes and images of the “chair” class with the same predicted view label in iterations 0, 100, 200, and 400 are shown in red boxes and blue boxes, respectively.

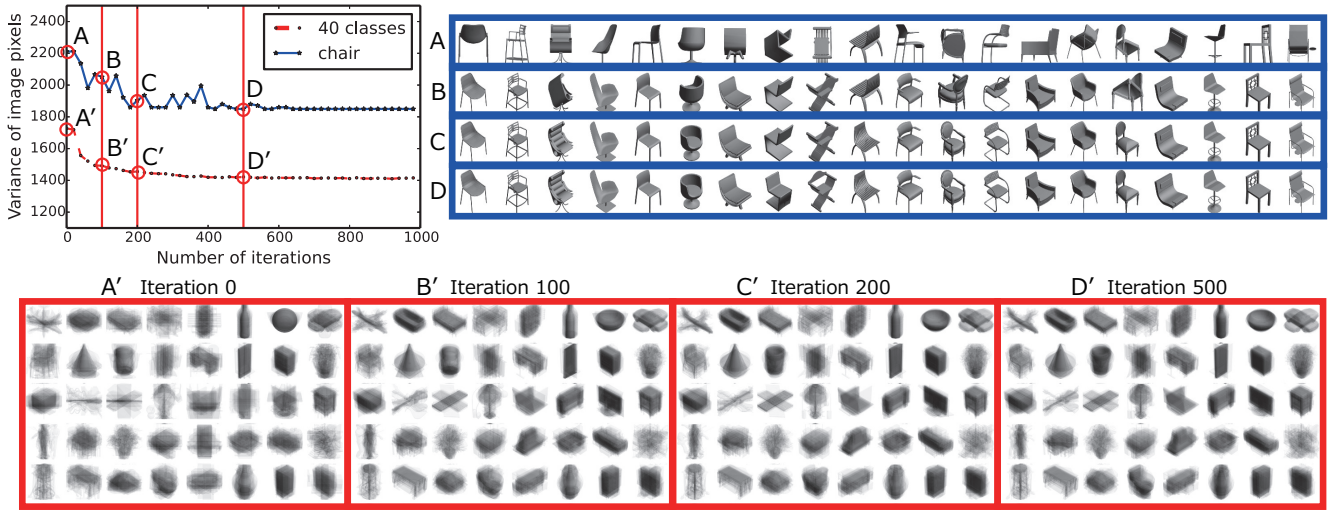


Figure 5. Variance change of the average images generated by concatenating multi-view images (case (ii)) in order of their predicted view labels. Average images of 40 classes and images of the “chair” class with the same predicted view label in iterations 0, 100, 200, and 500 are shown in red boxes and blue boxes, respectively.

recorded. We picked out 12 images of each object instance with the closest rotation angles to  $\{0^\circ, 30^\circ, \dots, 330^\circ\}$ . The variation of object classification accuracy w.r.t. the number of views used for prediction is shown in Fig. 9. The performance of RotationNet slightly surpassed the baseline method when all the 12 multi-view images were used for prediction. The reasons why the RotationNet is not very effective in this case are twofold. First, there is insufficient number of object instances per category; the least category has only two training samples. The baseline method is more robust to this issue than RotationNet because it regards im-

ages from all the viewpoints as the same class. Second, there are several cases where the upright assumption is actually invalid; the attitudes of object instances against the rotation axis are inconsistent in some object categories.

Table 2 summarizes the classification and view estimation accuracies. The baseline method and MVCNN are not able to estimate views because they are essentially view-point invariant. As another baseline approach to compare, we learned a CNN with AlexNet architecture that outputs 612 ( $= 51 \times 12$ ) scores to distinguish both viewpoints and categories, which we call “Fine-grained.” As shown in Ta-

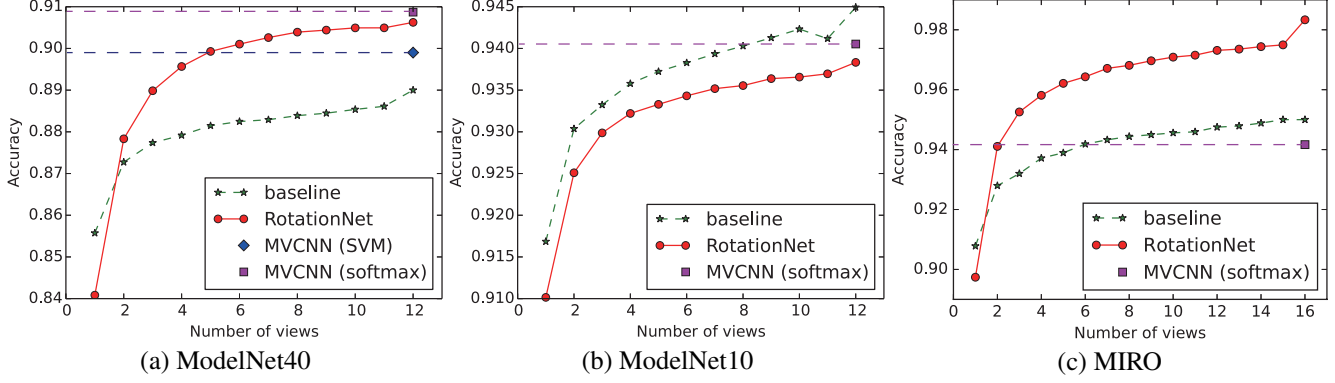


Figure 6. Classification accuracy vs. number of views used for prediction in case (i). From left to right are shown the results on ModelNet40, ModelNet10, and our new dataset MIRO.

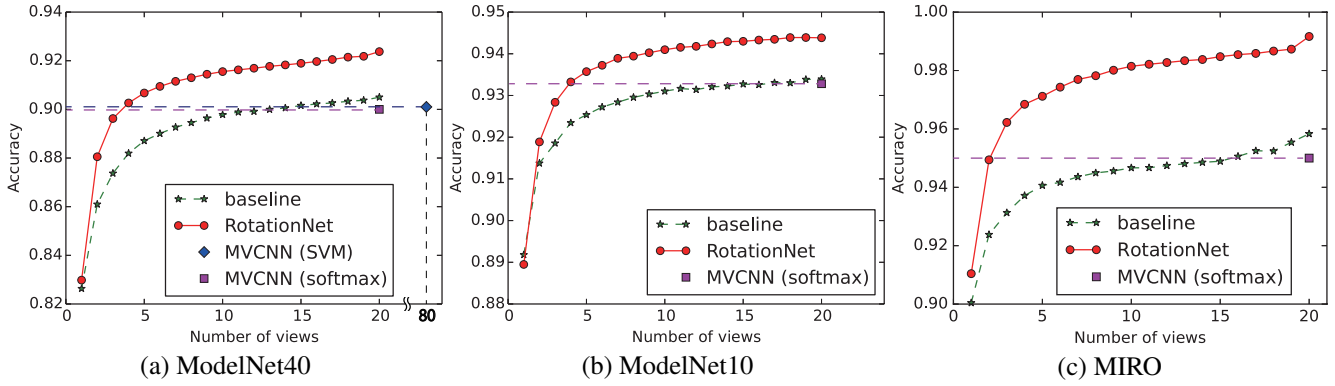


Figure 7. Classification accuracy vs. number of views used for prediction in case (ii). From left to right are shown the results on ModelNet40, ModelNet10, and our new dataset MIRO.

Algorithm	ModelNet40	ModelNet10
VRN Ensemble [4]	95.54	97.14
<b>RotationNet (Ours)</b>	92.38	94.38
ORION [20]	-	93.80
VRN [4]	91.33	93.61
FusionNet [8]	90.80	93.11
Pairwise [10]	90.70	92.80
MVCNN [23]	90.10	-
3D-GAN [25]	83.30	91.00
GIFT [1]	83.10	92.35
VoxNet [15]	83	92
DeepPano [21]	77.63	85.45
3DShapeNets [26]	77	83.50

Table 1. Comparison of classification accuracy (%). RotationNet achieved the *best* performance among the methods that do not take advantage of ensembling but used only a single model, both with ModelNet40 and ModelNet10.

ble 2, the classification accuracy with “Fine-grained” decreases while its viewpoint estimation accuracy improves as

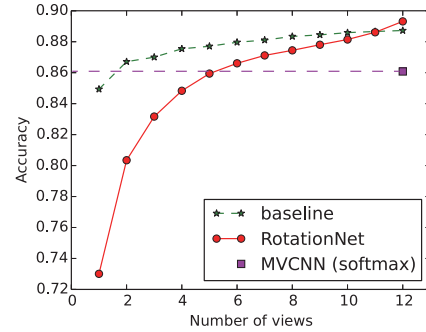


Figure 9. Classification accuracy vs. number of views used for prediction in case (i) with RGBD dataset.

the iteration grows. In contrast, RotationNet demonstrated the best performance in both object classification and viewpoint estimation, although the ground-truth poses are not given to RotationNet during the training.

Algorithm	class	view	Algorithm	class	view	Algorithm	class	view
MVCNN (softmax)	86.08	-	MVCNN (softmax)	94.17	-	MVCNN (softmax)	95	-
Baseline	88.73	-	Baseline	95	-	Baseline	95.83	-
Fine-grained, iter=300	81.23	26.94	Fine-grained, iter=800	92.76	56.72	Fine-grained, iter=1.1K	94.21	70.63
Fine-grained, iter=4K	76.95	31.96	Fine-grained, iter=4K	91.35	58.33	Fine-grained, iter=2.6K	93.54	72.38
<b>RotationNet</b>	<b>89.31</b>	<b>33.59</b>	<b>RotationNet</b>	<b>98.33</b>	<b>85.83</b>	<b>RotationNet</b>	<b>99.17</b>	<b>75.67</b>

Table 2. Accuracy of classification and view-point estimation (%) in case (i) with RGBD dataset

Table 3. Accuracy of classification and view-point estimation (%) in case (i) with MIRO dataset

Table 4. Accuracy of classification and view-point estimation (%) in case (ii) with MIRO dataset

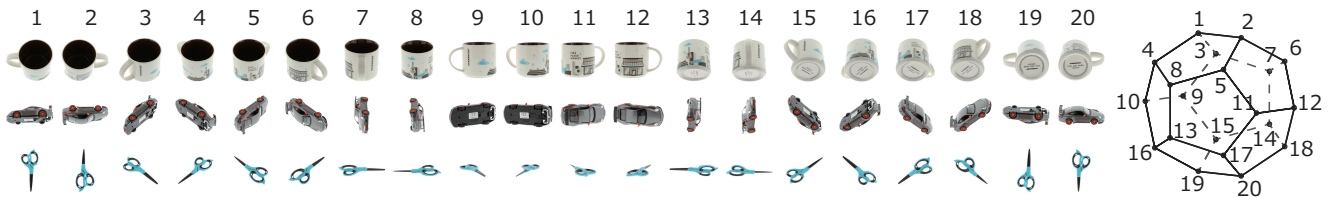


Figure 8. Exemplar multi-view images of objects in our new dataset MIRO. There are 20 images for each object instance in case (ii).

### 4.3. Experiment on a 3D rotated real image dataset

We describe the experimental results on our new dataset “Multi-view Images of Rotated Objects (MIRO)” in this section. We used Ortery’s 3D MFP studio<sup>3</sup> to capture multi-view images of objects with 3D rotations. As we mentioned in the previous section, the RGBD benchmark dataset [13] has two issues for training multi-view based CNNs: insufficient number of object instances per category (which is a minimum of two for training) and inconsistent cases to the upright assumption. Also, this dataset does not include the bottom faces of objects on the turning table. Our MIRO dataset includes 10 object instances per object category. It consists of 120 object instances in 12 categories in total. We captured each object instance with 10 levels of elevation angles and 16 levels of azimuth angles to obtain 160 images. For our experiments, we used 16 images ( $\theta = 22.5^\circ$ ) with  $0^\circ$  elevation of an object instance in case (i). We carefully captured all the object instances in each category to have the same upright direction in order to evaluate performance in the case (i). For case (ii), we used 20 images observed from the 20 vertices of a dodecahedron encompassing an object. Figure 8 shows examples of multi-view images of objects in case (ii).

Figures 6 (c) and 7 (c) show the object classification accuracy versus the number of views used for the prediction in case (i) and case (ii), respectively. In both cases, RotationNet clearly outperforms both MVCNN and the baseline method when the number of views is larger than 2. We also tested the “Fine-grained” method that outputs ( $192 =$

$12 \times 16$ ) scores in case (i) and ( $240 = 12 \times 20$ ) scores in case (ii) to distinguish both viewpoints and categories, and the overall results are summarized in Tables 3 and 4. Similar to the results with an RGBD dataset described above, there is a tradeoff between object classification and viewpoint estimation accuracies in the “Fine-grained” approach. RotationNet achieved the best performance in both object classification and viewpoint estimation, which demonstrates the strength of the proposed approach.

## 5. Discussions

We proposed RotationNet, which jointly estimates object category and viewpoint from each single-view image and aggregates the object class predictions obtained from a partial set of multi-view images. In our method, object instances are automatically aligned in an unsupervised manner with both inter-class and intra-class structures based on their appearance during the training. In the experiment using 3D object benchmark datasets ModelNet40 and ModelNet10, RotationNet outperformed the state-of-the-art methods that used a non-ensemble model, which are based on voxels and multi-view images. RotationNet is also able to achieve comparable performance to MVCNN [23] with 80 different multi-view images using only a couple of view images. Another contribution is that we developed a publicly available new dataset named MIRO, which contains multi-view images of 120 objects in 12 categories captured from 160 distinct viewpoints distributed in 3D space. Using this dataset and RGBD object benchmark dataset, we showed that RotationNet even outperformed a supervised learning based approach in a viewpoint estimation task. We consider

<sup>3</sup><https://www.ortery.com/photography-equipment/3d-modeling/3d-mfp/>



that our viewpoint estimation performance benefits from view-specific appearance information shared across classes due to the inter-class self-alignment.

Similar to MVCNN [23] and any other 3D object classification method that considers discrete variance of rotation, RotationNet has the limitation that each image should be observed from one of the pre-defined viewpoints. The next step is to introduce data augmentation with small variation of viewpoints to increase the robustness against viewpoint changes. Another potential venue to look into is employing an ensembling approach to further improve the accuracy. We expect that the ensemble of RotationNets with different architectures such as ResNet [7] and VGG-16 [22] boosts the performance, which we are interested in verifying.

## References

- [1] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. J. Latecki. Gift: A real-time and scalable 3d shape search engine. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [2] A. Bakry and A. Elgammal. Untangling object-view manifold for multiview recognition and pose estimation. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 434–449, 2014.
- [3] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Appearance-based active object recognition. *Image and Vision Computing*, 18(9):715–727, 2000.
- [4] A. Brock, T. Lim, J. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.
- [5] C.-Y. Chen and K. Grauman. Inferring unseen views of people. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2003–2010, 2014.
- [6] M. Elhoseiny, T. El-Gaaly, A. Bakry, and A. Elgammal. A comparative analysis and study of multiview cnn models for joint object categorization and pose estimation. In *Proceedings of International Conference on Machine Learning*, pages 888–897, 2016.
- [7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [8] V. Hegde and R. Zadeh. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016.
- [9] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 2017–2025, 2015.
- [10] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.
- [12] A. Kuznetsova, S. J. Hwang, B. Rosenhahn, and L. Sigal. Exploiting view-specific appearance similarities across classes for zero-shot pose prediction: A metric learning approach. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2016.
- [13] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2011.
- [14] K. Lai, L. Bo, X. Ren, and D. Fox. A scalable tree-based approach for joint object and pose recognition. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2011.
- [15] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.
- [16] L. Paletta and A. Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31(1):71–86, 2000.
- [17] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [19] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.
- [20] N. Sedaghat, M. Zolfaghari, and T. Brox. Orientation-boosted voxel nets for 3d object recognition. *arXiv preprint arXiv:1604.03351*, 2016.
- [21] B. Shi, S. Bai, Z. Zhou, and X. Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, 2015.
- [22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.
- [23] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 945–953, 2015.
- [24] H. Su, F. Wang, E. Yi, and L. J. Guibas. 3D-assisted feature synthesis for novel views of an object. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 2677–2685, 2015.
- [25] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2016.
- [26] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric

- shapes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.
- [27] H. Zhang, T. El-Gaaly, A. M. Elgammal, and Z. Jiang. Joint object and pose recognition using homeomorphic manifold analysis. In *Proceedings of AAAI Conference on Artificial Intelligence*, volume 2, page 5, 2013.
- [28] Z. Zhu, P. Luo, X. Wang, and X. Tang. Multi-view perception: a deep model for learning face identity and view representations. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 217–225, 2014.

## A. Candidates for view label variables $\{v_i\}_{i=1}^M$ in case (ii): w/o upright orientation

In the case where we do not assume upright orientation, we place virtual cameras on the  $M = 20$  vertices of a dodecahedron encompassing the object. There are three different patterns of rotation from a certain view, because three edges are connected to each vertex of a dodecahedron. Therefore, the number of candidates for all the view labels  $\{v_i\}_{i=1}^M$  is 60 ( $= 3M$ ). Figures 10-18 show all the candidates for a set of view label variables  $\{v_i\}_{i=1}^{20}$  in this case, in which vertex and image IDs are shown on the top and bottom rows respectively. Here,  $v_i$  indicates the ID of the vertex where the  $i$ -th image of the object instance is observed. For instance,  $\{v_i\}_{i=1}^{20}$  in Candidate #2 is  $\{1, 5, 2, 6, 3, 7, 4, 8, 13, 15, 14, 16, 17, 18, 19, 20, 9, 11, 10, 12\}$  (Fig. 10 (b)). The red star indicates the camera position where the first image is captured. The red dot indicates the camera position where the ninth image is captured.

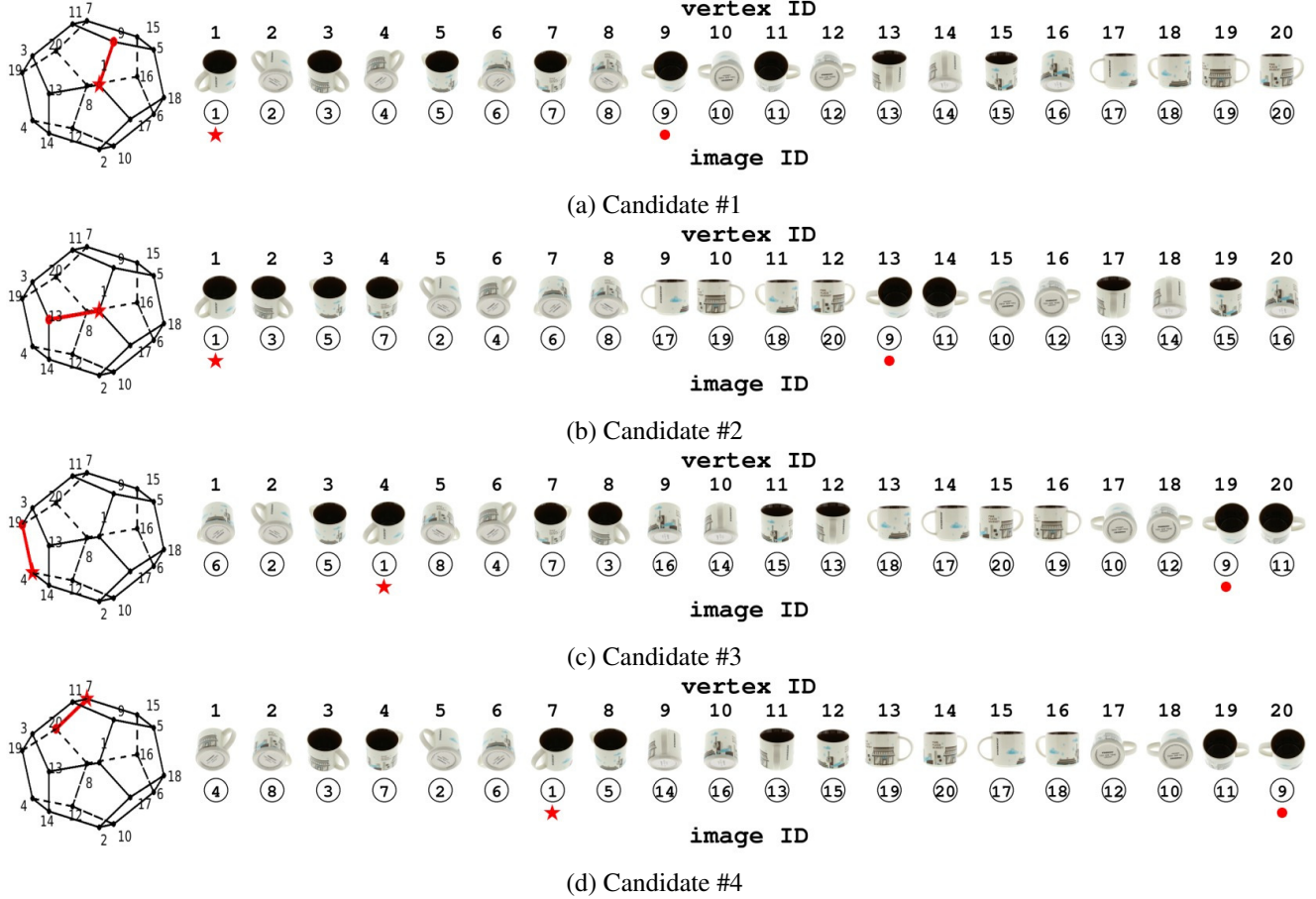


Figure 10. Candidates #1-#4 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.

## B. Object instances in MIRO dataset

Figure 19 shows thumbnail images of all the object instances in our new dataset MIRO. Our MIRO dataset includes 10 object instances per object category. It consists of 120 object instances in 12 categories in total. Each object instance has 160 images captured from different viewpoints approximately equally distributed in the spherical coordinates. An example of the multi-view images are shown in Fig. 20.

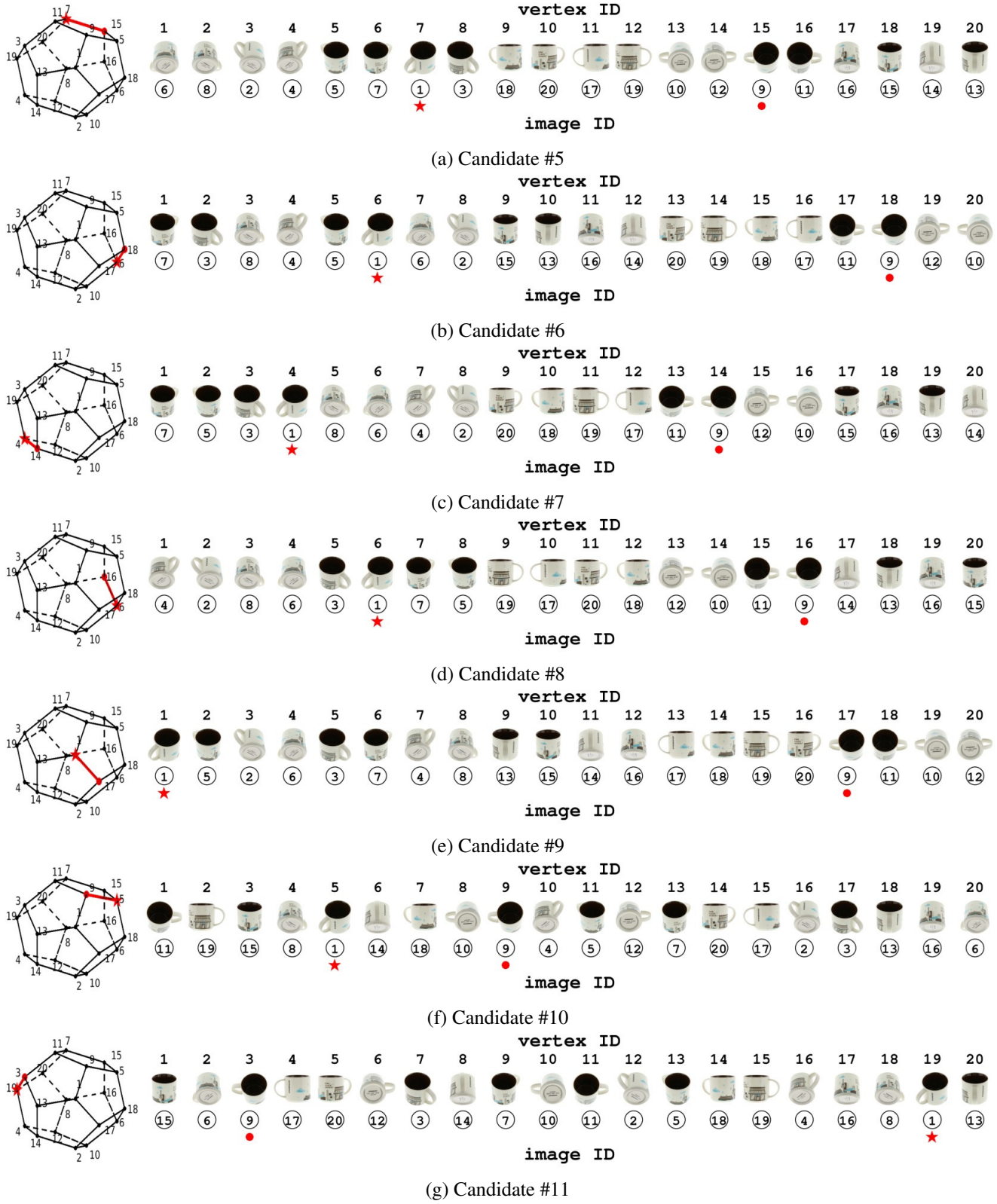


Figure 11. Candidates #5-#11 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.

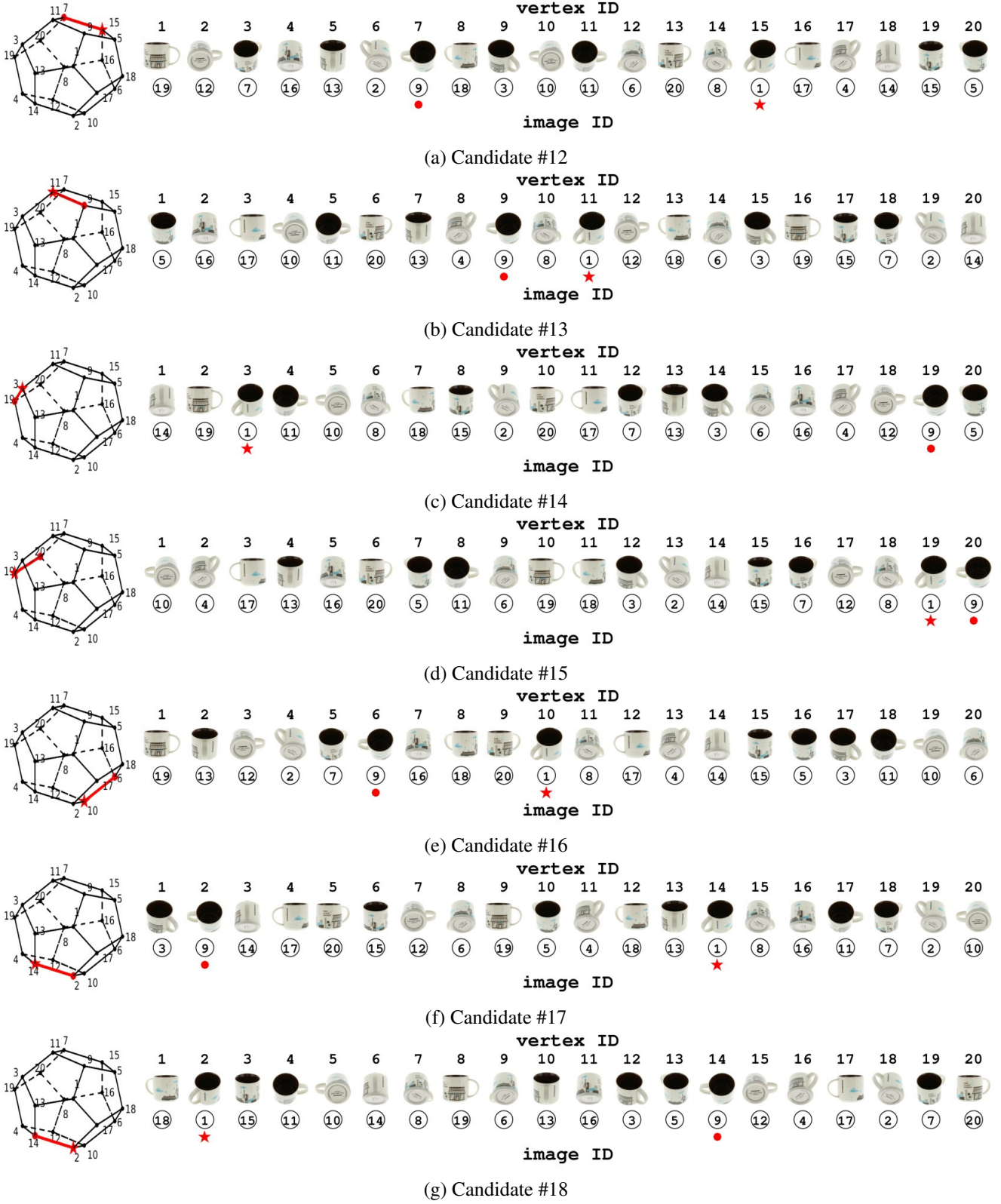


Figure 12. Candidates #12-#18 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.



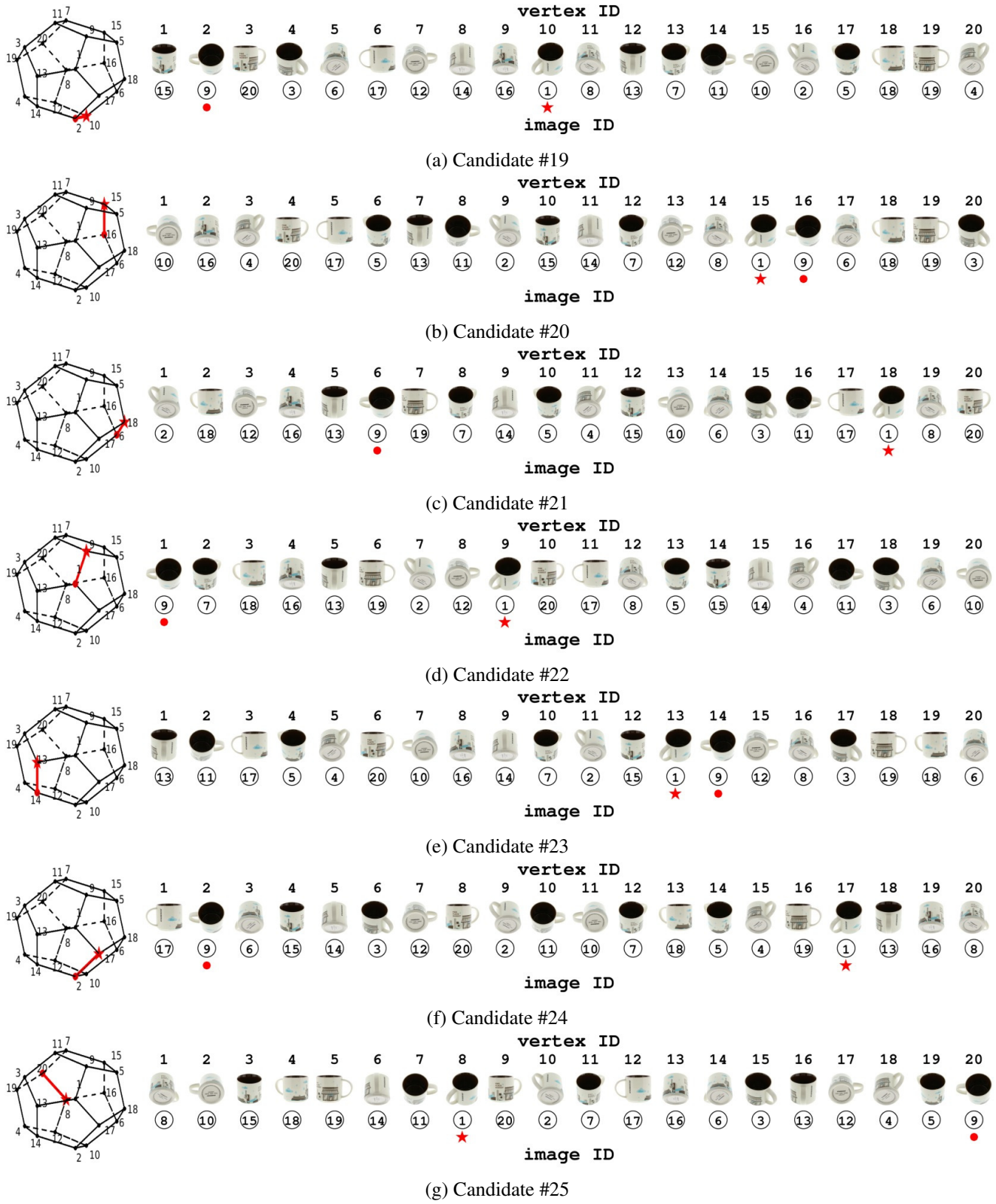


Figure 13. Candidates #19-#25 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.

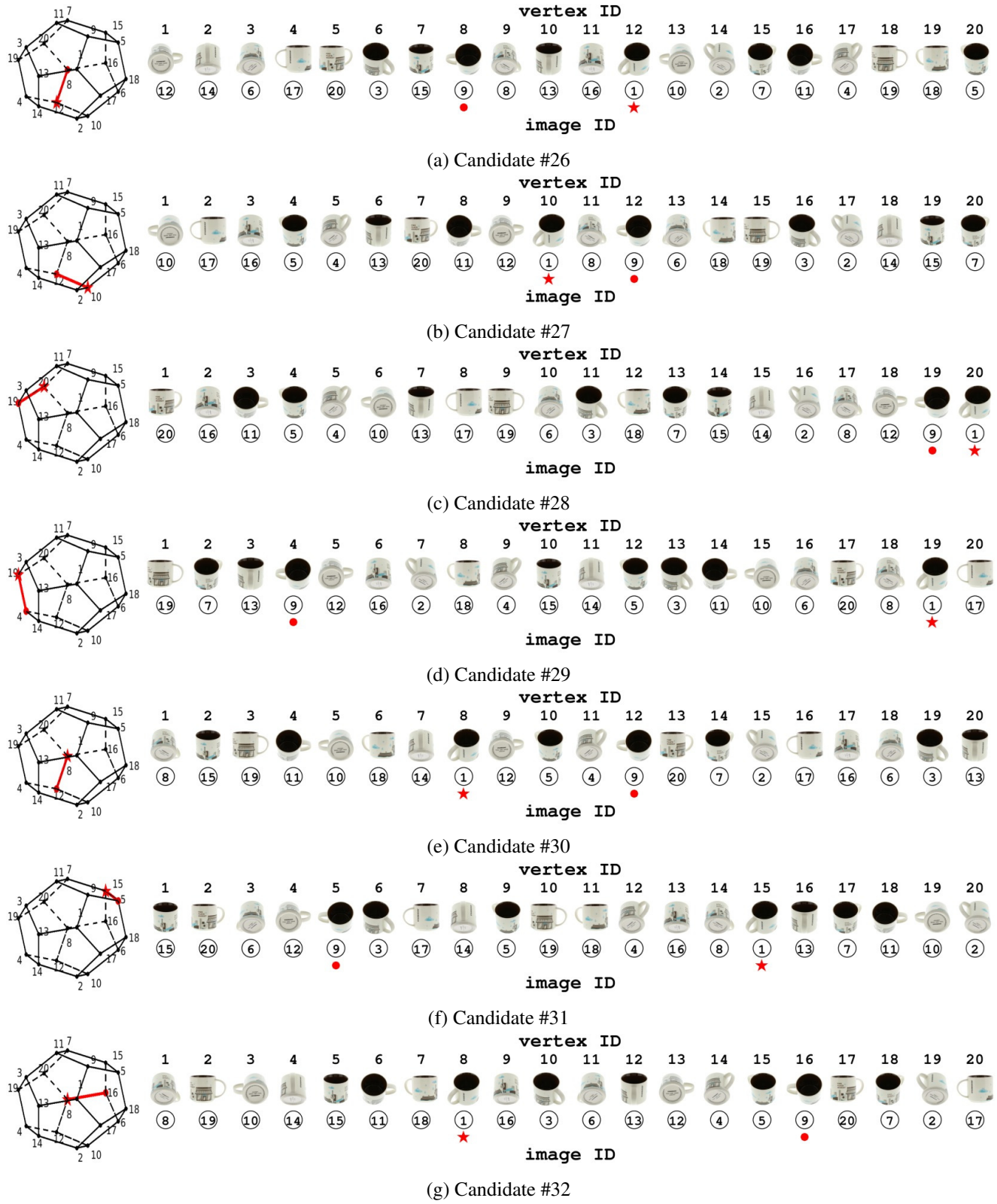


Figure 14. Candidates #26-#32 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.

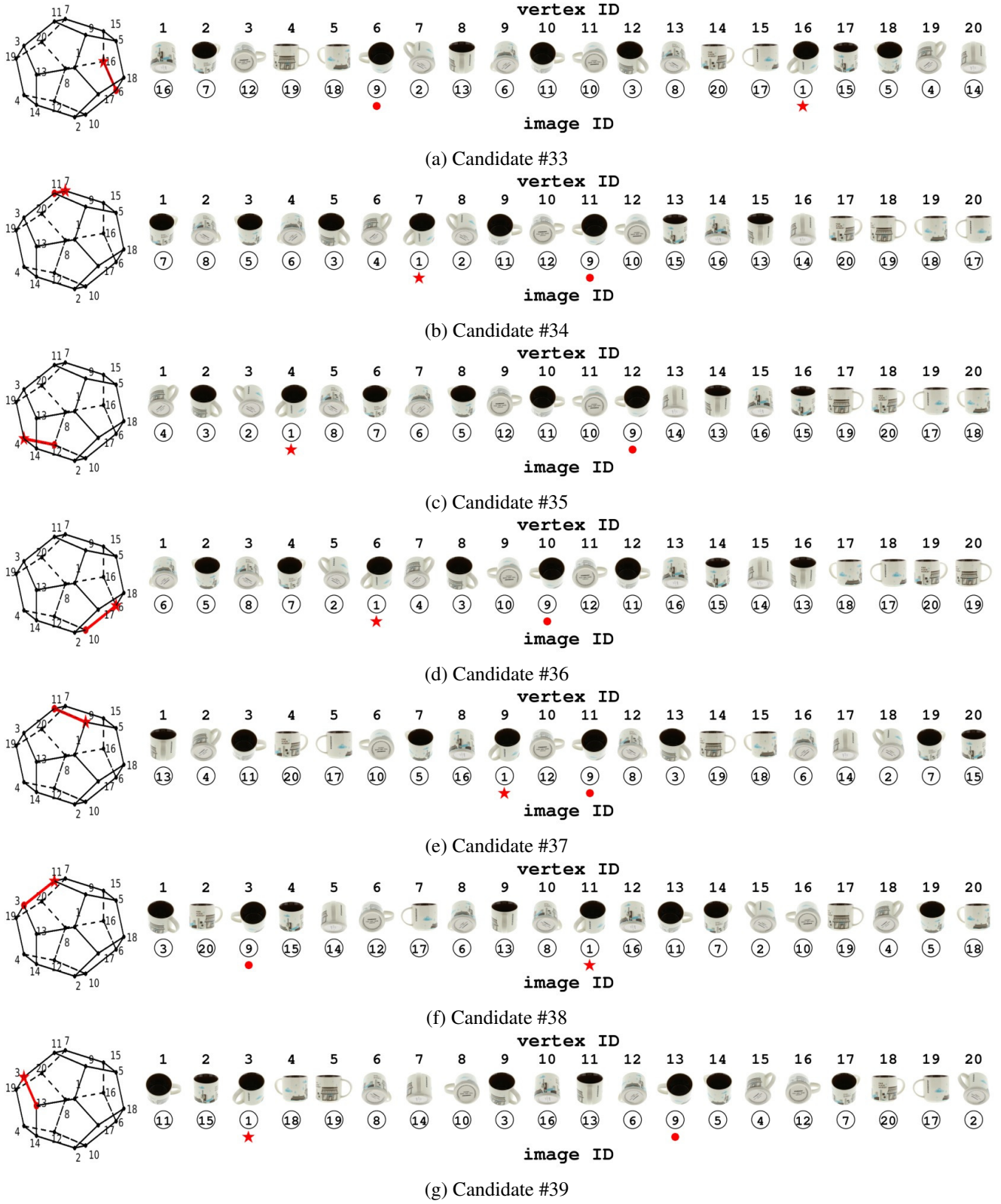


Figure 15. Candidates #33-#39 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.





(a) Candidate #40



(b) Candidate #41



(c) Candidate #42



(d) Candidate #43



(e) Candidate #44



(f) Candidate #45



(g) Candidate #46

Figure 16. Candidates #40-#46 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.

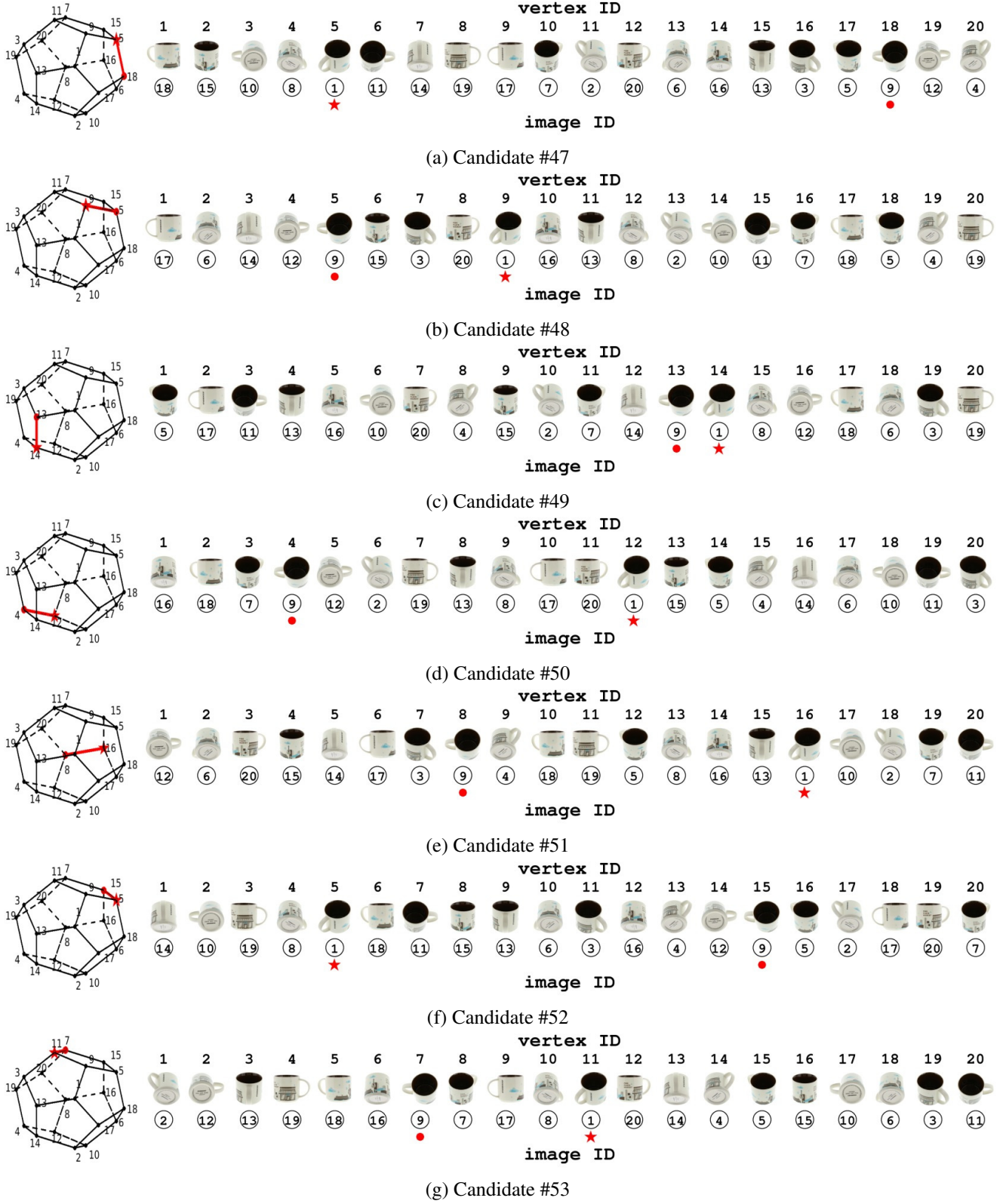


Figure 17. Candidates #47-#53 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.





(a) Candidate #54



(b) Candidate #55



(c) Candidate #56



(d) Candidate #57



(e) Candidate #58



(f) Candidate #59



(g) Candidate #60

Figure 18. Candidates #54-#60 for a set of view label variables  $\{v_i\}_{i=1}^{20}$  w/o upright orientation.



Figure 19. Thumbnail images of all the object instances in MIRO. From left to right are shown the object instances in “bus,” “car,” “cleanser,” “clock,” “cup,” “headphones,” “mouse,” “scissors,” “shoe,” “stapler,” “sunglasses,” and “tape\_cutter” category.

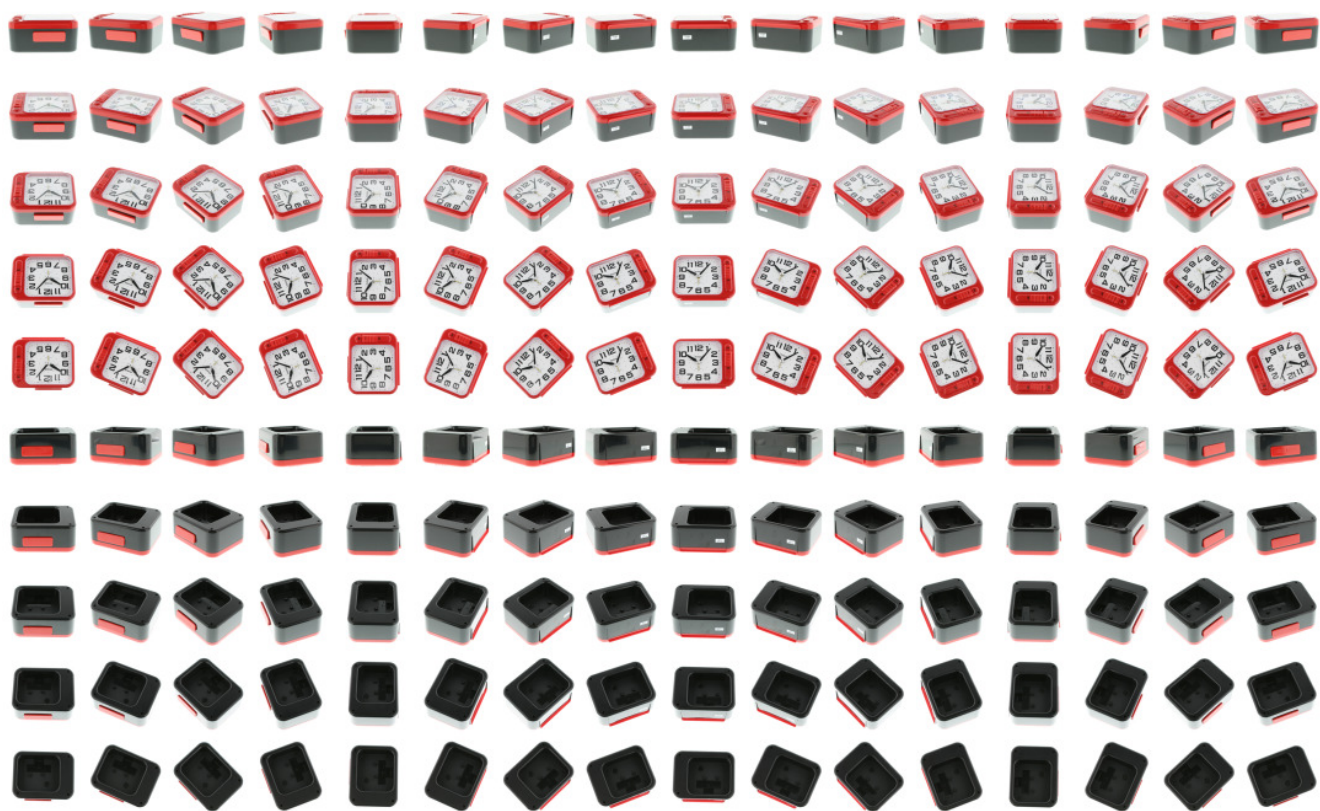


Figure 20. All the 160 images of an exemplar object instance captured from different viewpoints.